Appeared in: International Journal of Approximate Reasoning, Vol. 42, Nos. 1--2, pp. 101--118.

Sequential Influence Diagrams: A Unified Asymmetry Framework

Finn V. Jensen [a] Thomas D. Nielsen [a,*] Prakash P. Shenoy [b]

[a] *Department of Computer Science, Aalborg University, Fredrik Bajers vej 7E, 9220 Aalborg Ø, Denmark*

[b] *School of Business, University of Kansas, 1300 Sunnyside Ave, Summerfield Hall, Lawrence, KS 66045-7585 USA*

Abstract

We describe a new graphical language for specifying asymmetric decision problems. The language is based on a filtered merge of several existing languages including sequential valuation networks, asymmetric influence diagrams, and unconstrained influence diagrams. Asymmetry is encoded using a structure resembling a clustered decision tree, whereas the representation of the uncertainty model is based on the (unconstrained) influence diagram framework. We illustrate the proposed language by modeling several highly asymmetric decision problems, and we describe an efficient solution procedure.

Key words: Asymmetric decision problems, (asymmetric) influence diagrams, sequential valuation networks, unconstrained influence diagrams.

1 Introduction

There are mainly two popular classes of graphical languages for representing sequential decision problems with a single decision maker, namely decision trees (DTs) [1] and influence diagrams (IDs) (including valuation networks (VNs)) [2,3]. Decision trees are very expressive, but the specification load, i.e., the size of the graph, increases exponentially with the number of decisions and observations. This means that the specification load becomes intractable even for medium sized decision problems. On the other hand, the specification load for IDs increases linearly in the number of decisions and observations, but the expressiveness of IDs is limited.

* Corresponding author.
Email addresses: fvj@cs.aau.dk (Finn V. Jensen), tdn@cs.aau.dk (Thomas D. Nielsen), pshenoy@ku.edu (Prakash P. Shenoy).

Many attempts have been made to reduce the specification load for decision trees, for example coalesced DTs [4], but so far they do not seem to have made a substantial impact. Other researchers work on extending the scope of IDs. The basic limitation of IDs is that they can only (efficiently) represent symmetric decision problems: a decision problem is said to be *symmetric* if i) in all of its decision tree representations, the number of scenarios is the same as the cardinality of the Cartesian product of the state spaces of all chance and decision variables, and ii) in one decision tree representation, the sequence of chance and decision variables is the same in all scenarios. A decision problem is said to be *asymmetric* if it is not symmetric.

One line of extending the scope is to introduce features for representing asymmetric decisions problems [5–14]. A special aspect of asymmetric decision problems is that the next observation or decision may depend on the past. This means that not only is the outcome of the decision or observation dependent of the past, but so is the very observation. If, for example, you have the option of going to a movie or to a restaurant, then tasting the meal is meaningless if you have decided to go to the movie. Another issue, which for some time has been overlooked, is that the order of decisions and observations may not be settled and it is therefore part of the decision problem. If you for example have two tests and two treatments for a disease, then a strategy is not a plain sequence of tests and treatments, but rather a conditional sequence representable by a directed acyclic graph, where the different paths correspond to different orderings of the decisions and observations [15]. To distinguish between the two types of asymmetry, we shall talk about *structural asymmetry* and *order asymmetry*.

Recently, two frameworks for representing asymmetric decision problems have been proposed in [12,14]. In the asymmetric influence diagram (AID) [12], the model is based on a Bayesian network extended with features for representing decisions and utilities. Thus, we may have chance nodes, which are neither observed during the decision process nor do they appear in the domain of a utility function, but they are still included in the model since they play a role as mediating the probabilities. On the other hand, in the sequential valuation network (SVN) [14], the model is based on a compact representation of a DT. This means that mediating variables are usually not considered part of the actual decision problem, and they are therefore marginalized out during the modeling phase; the probability potentials need not be conditional probabilities.

In the present paper we merge and filter the various suggestions (in particular, the two approaches mentioned above), into one language called *sequential influence diagrams* (SIDs). In the proposed language we have an explicit Bayesian network representation of the uncertainty model, and also an explicit representation of the sequencing of decisions and observations using a

structure, related to that of SVNs, that allows for structural as well as order asymmetry. The syntax and semantics of the SID representation supports an efficient solution procedure, where the original asymmetric decision problem is decomposed into a collection of symmetric sub-problems that can be solved recursively. The solutions to the "smaller" symmetric sub-problems then constitute a solution to the original asymmetric decision problem.

2 Some Examples

In this section we present three examples to illustrate the types of asymmetry discussed above. These examples will also subsequently be used to introduce some of the features of the proposed framework.

2.1 Structural asymmetry: The reactor problem

The Reactor problem was originally described in [9]. Here we will describe the adaptation proposed in [10]. An electric utility firm must decide whether to build (B) a reactor of advanced design (a), a reactor of conventional design (c), or no reactor (n) at all. If the reactor is successful, i.e., there are no accidents, an advanced reactor is more profitable, but it is also riskier: If the firm builds a conventional reactor, the profits are \$8B if it is a success (cs), and −\$4B if there is a failure (cf). If the firm builds an advanced reactor, the profits are \$12B if it is a success (as), −\$6B if there is a limited accident (al), and −\$10B if there is a major accident (am). The firm's utility is assumed to be linear in dollars. Before making the decision to build, the firm has the option to conduct a test ($T = t$) or not (nt) of the components of the advanced reactor. The test results (R) can be classified as either bad (b), good (g), or excellent (e). The cost of the test is \$1B. The test results are highly correlated with the success or failure of the advanced reactor (A), and the strength of the correlation is encoded in the causal probability model shown in Fig. 1(a). If the test results are bad, then the Nuclear Regulatory Commission (NRC) will not permit the construction of an advanced reactor. A curious aspect of this problem is that if the firm decides not to conduct the test (and it is not required to do so by the NRC), it can proceed to build an advanced reactor without any constraints from the NRC. Fig. 1(b) shows a decision tree representation of this problem, where the probabilities can be found from the probability model in Fig. 1(a).

The specification of the quantitative part (which includes the probability model) can actually be further separated from the decision tree. For example, Fig. 1(c) depicts a structure containing all the functions involved in the specification of the decision problem; the diamond shaped nodes are utility nodes and their parents are the arguments in the associated utility functions (as in the DT, the decision nodes are drawn as rectangles). However, by comparing the specification in Fig. 1(c) with the one in Fig. 1(b), we notice that neither

the order of the decisions and observations nor the asymmetry constraints are specified. In the proposed framework, this is done through so-called structural arcs with annotations (see Fig. 3), which will be described further in Section 3.

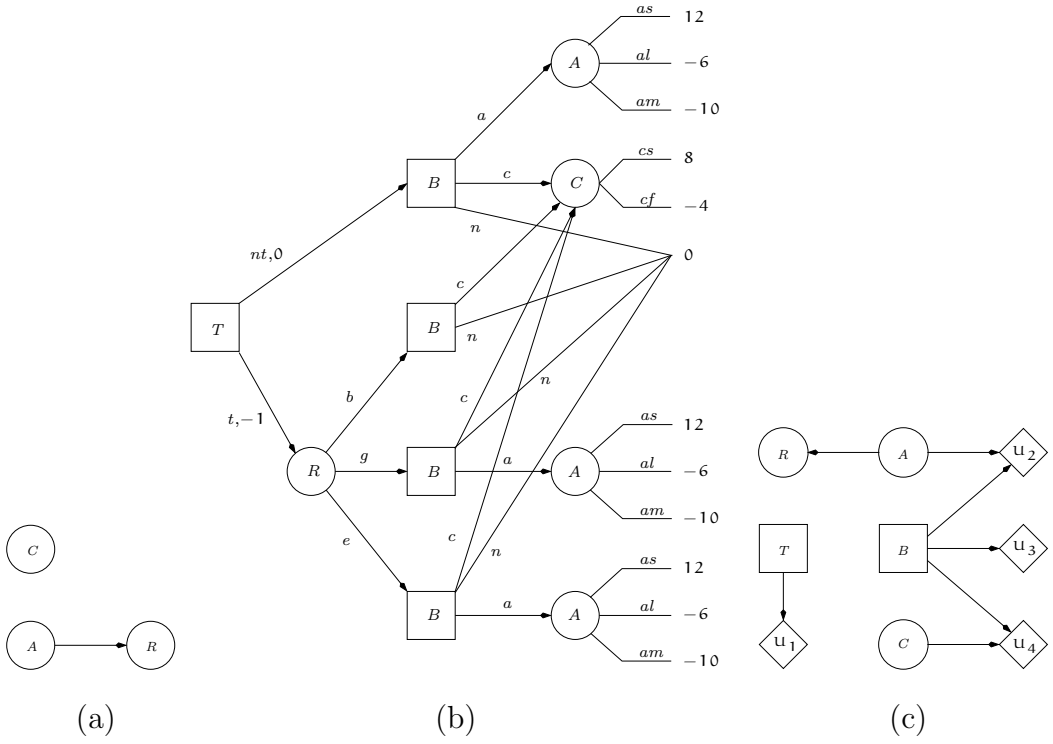


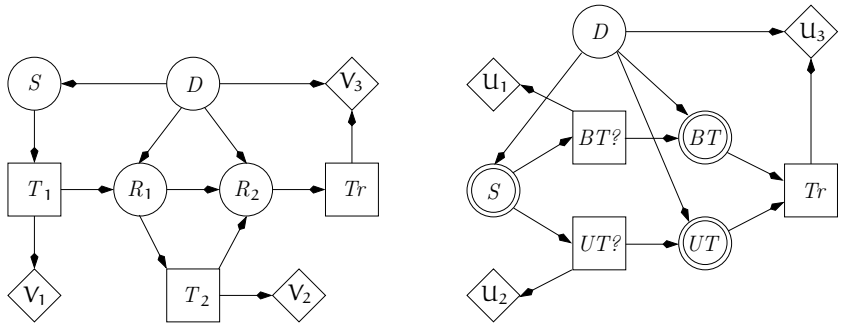
Fig. 1. Figure (a) shows a causal model that provides the required probabilities for the decision tree representation of the REACTOR PROBLEM shown in figure (b). Figure (c) shows a further refinement of the quantitative model for the DT, but it lacks information about order and asymmetry constraints.

2.2 Order asymmetry: The diagnosis problem

A physician is trying to decide on a policy for treating patients, which after an initial examination of their symptoms (S) are suspected to suffer from diabetes (D). Diabetes has two symptoms, glucose in urine and excessive glucose in blood. Before deciding on whether or not to treat for diabetes, the physician can decide to perform a blood test ($BT?$) and/or a urine test ($UT?$) which will produce the test results BT and UT, respectively. After the physician has observed the test results (if any) she has to decide whether to treat the patient for diabetes. Observe that in this decision problem the sequence in which the tests are decided upon is unspecified, and that the test result of e.g. the blood test is only available if the physician actually decides to perform the test; similarly for the result of the urine test.

To represent this problem by an influence diagram we have to represent the unspecified ordering of the tests as a linear ordering of decisions. This can be done by introducing two decision variables, T_1 and T_2 as shown in Fig. 2(a);

the two variables model the decisions concerning the first test and the second test, respectively. We assume that the uncertainty attached to the tests is so, that repeating a test will give the same test result. This is represented by the arc $R_1 \rightarrow R_2$. Unfortunately, the structure of the decision problem is not apparent from the model and, for large decision problems, this modeling technique will be prohibitive as all possible scenarios should be explicitly encoded in the model. As an alternative, [15] describes the unconstrained influence diagram (UID) for representing these types of decision problems. In the UID the combinatorial problem of representing the possible decision scenarios is postponed to the solution phase; Fig. 2(b) depicts a UID representation of the DIAGNOSIS PROBLEM.



(a) (b)

Fig. 2. Figure (a) shows an ID representation of the DIAGNOSIS PROBLEM. The test decisions (T_1 and T_2) have three states, *bt*, *ut* and *no-test*, and the result-nodes (R_1 and R_2) have five states, pos_{bt}, neg_{bt}, pos_{ut}, neg_{ut} and *no-result*. The arc $R_1 \rightarrow R_2$ encodes that a repeated test will give an identical result. Figure (b) shows the corresponding UID representation. The doubled circled nodes are observed when all their temporal predecessors have been observed; the nodes BT and UT have an extra state, *no-result*.

In the proposed framework we partly adopt the UID representation by introducing *clusters* of nodes. A cluster is a part of the model for which the ordering of the decisions and observations is only partly specified (more details in Section 3). Fig. 4(b) depicts the SID representation of the DIAGNOSIS PROBLEM.

2.3 Structural as well as order asymmetry: The dating problem

Joe needs to decide whether he should ask (*Ask?*) Emily for a date for Friday evening. He is not sure if Emily likes him or not (*LikesMe*). If he decides not to ask Emily or if he decides to ask and she turns him down, he will then decide whether to go to a nightclub or watch a movie on TV at home (*NClub?*). Before making this decision, he will consult the TV guide to see if there are any movies he would like to see (TV). If he decides to go to a nightclub, he will have to pay a cover charge and pay for drinks. His overall nightclub experience (*NCExp*) will depend on whether he meets his friends (*MeetFr*),

the quality of the live music, etc (*Club*). If Emily accepts (*Accept*), then he will ask her whether she wishes to go to a restaurant or to a movie (*ToDo*); Joe cannot afford to do both. If Emily decides on a movie, Joe will have to decide (*Movie*) whether to see an action movie he likes or a romantic movie that he does not really care for, but which may put Emily in the right mood (*mMood*) to enhance his post-movie experience with Emily (*mExp*). If Emily decides on a restaurant, he will have to decide (*Rest.*) on whether to select a cheap restaurant or an expensive restaurant. He knows that his choice will have an impact on his wallet and on Emily's mood (*rMood*) that in turn will affect his post-restaurant experience with Emily (*rExp*).

3 Sequential Influence Diagrams

In this section we will describe the main features of sequential influence diagrams (SIDs) by considering the SID representation of the REACTOR PROBLEM, the DIAGNOSIS PROBLEM and the DATING PROBLEM as described in the previous section.

An SID can basically be seen as two diagrams superimposed onto each other. One diagram encodes information precedence as well as structural and order asymmetry, whereas the other encodes functional relations for the utility nodes $\mathcal{U}_{\mathsf{V}}$ (drawn as diamonds) and probabilistic dependence relations for the chance nodes $\mathcal{U}_{\mathsf{C}}$ (drawn as ellipses); following the standard convention we depict decision nodes $\mathcal{U}_{\mathsf{D}}$ using rectangles (see Fig. 3). The set of all nodes/variables is denoted $\mathcal{U} = \mathcal{U}_{\mathsf{V}} \cup \mathcal{U}_{\mathsf{C}} \cup \mathcal{U}_{\mathsf{D}}$.

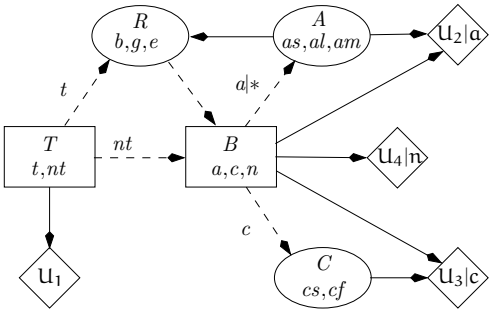


Fig. 3. An SID representation of the REACTOR PROBLEM; the $*$ denotes that the choice $B = a$ is only allowed in scenarios that satisfy $(T = nt) \vee (T = t \wedge (R = e \vee R = g))$.

Dashed arrows (called *structural arcs*) encode the structure of the decision problem, i.e., information precedence and asymmetry; $\mathsf{X} \dashrightarrow \mathsf{Y}$ means that Y is observed/decided upon after X has been observed/decided upon. A structural arc (X, Y) may be associated with an annotation $\mathsf{g}_{(\mathsf{X},\mathsf{Y})}$ (called a *guard*) consisting of two parts. The first part describes the condition under which the next node in the set of scenarios is the node that the arc points to; when the condition is fulfilled we say that the arc is *open*. For example, in Fig. 3, the

annotation t on the dashed arc from T to R means that whenever $T = t$, the next node in all scenarios is R. If there are constraints on the choices at any decision node, then this is specified in the second part of the annotations. The choices at T are unconstrained, hence the annotations on all edges emanating from T have only one part. On the other hand, the choice $B = a$ is only allowed in scenarios that satisfy $(T = nt) \vee (T = t \wedge (R = e \vee R = g))$, and this is indicated by the second part of the annotation on the arc from B to A. The set of nodes referenced by a guard g is called the domain of g, e.g. $\mathrm{dom}\left(\mathsf{g}_{(B,A)}\right) = \{T, R, B\}$. The set of annotations/guards is denoted $\mathcal{G}$ and in the remainder of this paper we shall assume that all structural arcs are associated with a guard, i.e., if $\mathcal{G}$ does not contain a guard for the structural arc (X, Y), then we extend $\mathcal{G}$ with the guard $\mathsf{g}_{(\mathsf{X},\mathsf{Y})} \equiv \mathsf{1}|\mathsf{1}$ (although this will not be shown explicitly in the models).

The set of scenarios defined by an SID can be identified by iteratively following the open arcs from a source node (a node with no incoming structural arcs) until a node is reached with no open outgoing arcs; note that we do not require a unique source node, and as we shall see later, the structure of an SID ensures that we have a finite number of scenarios and that each scenario has a finite number of elements.

From the description above, we note that a scenario does not require an explicit representation of the terminal node. Thus in cases $B = a$, the scenarios end with a state of A, if $B = c$, the scenarios end with a state of C, and if $B = n$, then the scenarios end at B. The solid arcs that point to chance and utility nodes have the same meaning as in IDs, i.e., these arcs encode the structure of the probability and utility model for the decision problem (note that we do not allow annotations to be associated with these arcs). Similar to the frameworks described in [12,14], we advocate the use of partial probability and utility potentials to emphasize the conceptual distinction between a configuration with zero probability and a logically inconsistent configuration. This also reduces the specification load as illustrated in Fig. 3, where the utility potential $\mathsf{U}_2|\mathsf{a}$ is only specified for $B = a$. The standard operations of combination and marginalization are readily extended to these types of potentials by treating the undefined value as an additive identity and a multiplicative zero. Following the usual notation we denote the set of probability and utility potentials by Φ_{I} and Ψ_{I}, respectively.

In the reactor problem, all chance nodes appear in some scenarios. However, this may not always be the case. In the sequential influence diagram for the DATING PROBLEM (see Fig. 4(a)), we have several chance nodes (i.e., *LikesMe*, *mMood*, *rMood*) that do not appear in any scenario. However, we still include these variables in the representation since the probability distribution of the chance variables, that do appear in a scenario, are influenced by these chance variables; note that in the SVN framework these variables would have

been marginalized out. In general we distinguish between observable and non-observable variables; a variable X is said to be *observable* if there is at least one decision scenario in which the true state of X is observed by the decision maker. Syntactically we identify the observable nodes as the set (denoted $\mathcal{U}^{\mathrm{o}}$) of nodes associated with a structural arc; analogously, the set of unobserved nodes $\mathcal{U} \setminus \mathcal{U}^{\mathrm{o}}$ is denoted $\mathcal{U}^{\mathrm{u}}$. This also means that an observable chance node may be connected to both a solid and a dashed arc originating from the same node, say Y; semantically, this implies that the chance node is not only observed after Y, but it is also probabilistically dependent on Y. This is for instance the case with the variables *Accept* and *ToDo* in Fig. 4(a).

3.1 Partial Temporal Orderings

From the description above we see that the part of the SID that encodes structural asymmetry is closely related to sequential decision diagrams (SDDs) [9] and clustered decision trees [4]. Unfortunately, this also implies that the proposed language inherits some of the limitations associated with these representation languages. For instance, if only a partial temporal ordering exists for e.g. a set of chance nodes, then we need to impose an artificial linear ordering on these nodes. Note that although a partial temporal ordering over chance nodes is of no importance when looking for an optimal strategy (see Section 4), it may still be important when considering the SID framework as a tool for communication.

In order to extend the expressive power of the proposed language, we introduce a syntactical construct in the form of a *cluster* of nodes: in terms of information precedence, we can think of a cluster $\mathbf{C}$ of nodes as a single node in the sense that a structural arc going into $\mathbf{C}$ from a node X indicates that after X has been observed or decided upon the next node is a node in $\mathbf{C}$. A structural arc from $\mathbf{C}$ to a node Y indicates that Y will be the next node in the ordering when leaving $\mathbf{C}$. Fig. 4(a) illustrates the use of clusters for representing the partial temporal ordering over the chance nodes *Club* and *MeetFr* in the DATING PROBLEM; the cluster is depicted by a dotted ellipse. From the model we see that these two nodes will only be observed by the DM after deciding on *NClub?* but before observing *NCExp*.

The example above illustrates how unspecified temporal orderings over chance nodes may be represented in the SID framework using clusters. However, unspecified/partial temporal orderings may be more complicated as it can also relate to orderings of decisions and observations. For instance, in the DIAGNOSIS PROBLEM, the DM has to decide on whether to perform a blood test (*BT?*) and/or a urine test (*UT?*), but the order in which the decisions are made is unspecified. This type of decision problem is usually modeled by introducing two decision nodes, T_1 and T_2, representing the decision on the first test and the second test respectively. I.e., T_1 would have the states *bt* (blood

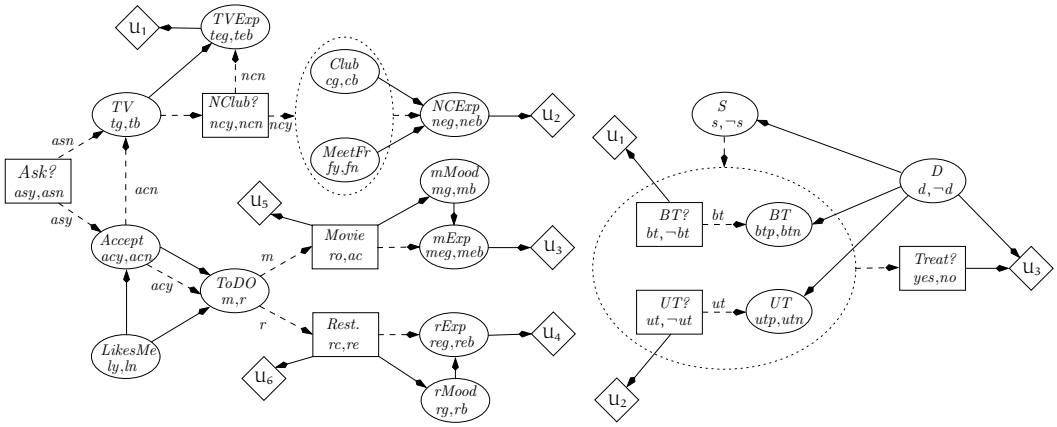


Fig. 4. Figure (a) shows an SID representation of the DATING PROBLEM. Figure (b) shows an SID representation of the DIAGNOSIS PROBLEM. The structural arc emanating from the cluster indicates that any decision scenario within the cluster is followed by a decision on *Treat?*.

test), *ut* (urine test) and *no-test*; similarly for T_2. Unfortunately, this technique will (in standard representation languages such as IDs) require either dummy variables or dummy states due to the asymmetric nature of the information constraints, e.g., if $\mathsf{T}_2 = bt$, then BT is observed before deciding on T_2 whereas UT is unobserved (conversely if $\mathsf{T}_1 = ut$). That is, we basically need to include all admissible decision/observation sequences directly in the model (see also Section 2.2). In order to avoid this problem we advocate the approach proposed in [15].[1] That is, instead of making the possible decision sequences explicit in the model (through nodes like T_1 and T_2) we postpone it to the solution phase by allowing the temporal ordering to be unspecified; note that this also implies that when solving the SID we not only look for an optimal strategy for the decisions but also for an optimal conditional ordering of the decisions. For example, Fig. 4(b) depicts the SID representation of the DIAGNOSIS PROBLEM, where the ordering of the decisions *BT?* and *UT?* (as well as the corresponding results) is unspecified.

In this model we have a cluster with a partial ordering over the nodes *BT?*, BT, *UT?* and UT. The ordering specifies that $BT? \prec BT$ and that the result of the blood test, BT, is only revealed if we initially decide to have the blood test performed, $BT? = bt$; similar for *UT?* and UT. Observe that the set of decision scenarios encoded in the cluster corresponds to a collection of conditional extensions of the partial ordering that produces total orderings.

Finally, it should be emphasized that the SID framework allows for the specification of directed (temporal) cycles, with the restriction that before any

[1] Note that the unconstrained ID focuses only on order asymmetry (not asymmetry in general) and therefore requires a limited use of dummy states and/or variables.

of the nodes in the cycle are observed the cycle must be "broken". That is, the cycle should contain at least one closed structural arc. The use of cycles supports the specification of decision/observation sequences that depend on previous observations and decisions. For example, consider a doctor who has just performed a test on a patient but has not yet received the test results (R). There is a risk that the test results will be delayed (D) for a day or two, but before the end of the day she must decide on a treatment (Tr) for the patient. A partial SID representation of this scenario is shown in Fig. 5, where the conditional temporal ordering of the observation of the test result and the decision on the treatment is modeled using a directed cycle.

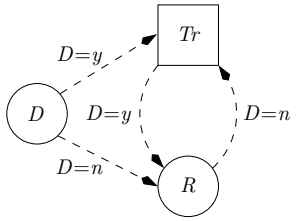


Fig. 5. The cycle models that the temporal ordering of Tr and R depends on the state of D.

In general, an SID I induces a *partial temporal order* $\prec_{\mathrm{I}}$ on the nodes in $\mathcal{U}_{\mathrm{C}} \cup \mathcal{U}_{\mathrm{D}}$, i.e., $\mathsf{X} \prec_{\mathrm{I}} \mathsf{Y}$ if and only if Y is either unobserved or there exists a directed path (consisting of structural arcs) from X to Y in I but not from Y to X. For the example above, we see that Tr and R are incomparable before the observation of D. However, by observing D the cycle is "broken" and Tr and R become comparable under $\prec_{\mathrm{I}}$.

4 Solution

When solving an SID we not only look for an optimal policy for each decision variable but also for an optimal sequencing of the variables (when order asymmetry is present). More specifically, a solution to an SID includes a collection of *step-functions* specifying the next observation/decision given the current information. I.e., for each node X we look for a function $\sigma_{\mathsf{X}} : \mathrm{sp}\,(\mathrm{past}(\mathrm{X})) \rightarrow \mathrm{Succ}(\mathrm{X})$, where $\mathrm{sp}\,(\mathrm{past}(\mathrm{X}))$ is the state space of the variables, $\mathrm{past}(\mathsf{X})$, that occur before X in the temporal ordering and $\mathrm{Succ}(\mathsf{X})$ is the set of possible immediate temporal successors of X. Observe that when order asymmetry is not present, then $|\mathrm{Succ}(\mathsf{X})| = 1$ and σ_{X} is therefore trivial. In this special case the SID can in principle be solved by unfolding it into a decision tree and then using the average-out and fold-back algorithm [1] on that tree; inspired by [5], the required probabilities can be calculated from the probability potentials specified by the realization of the SID. When the SID contains order asymmetry we can apply the same technique except that we now construct a family of decision trees, one for each possible configuration of the variables subject to order asymmetry. An optimal strategy can then be found in the

DT with maximum expected utility, and from the structure of this decision tree we have a specification of the set of optimal step functions.

4.1 Decomposition graphs

Unfortunately, the brute force approach described above has a tendency to create unnecessarily large decision trees in case the original decision problem contains symmetric sub-structures. In order to overcome this shortcoming, we propose a solution technique resembling that for sequential valuation networks and asymmetric influence diagrams. That is, we basically (i) decompose the asymmetric decision problem into a collection of symmetric subproblems organized in a so-called *decomposition graph*, and (ii) propagate probability and utility potentials upwards from the leaves. The decomposition graph is constructed by following the temporal ordering and recursively instantiating the so-called *split variables* w.r.t. their possible states.

Definition 4.1 *Let* I *be an SID with variables* $\mathcal{U}$ *and guards* $\mathcal{G}$*. A variable* $\mathsf{X} \in \mathcal{U}$ *is said to be a* split variable *if it appears in the domain of a guard, i.e., if there exists a* $\mathsf{g} \in \mathcal{G}$ *s.t.* $\mathsf{X} \in dom(\mathsf{g})$*.*

A split variable X is said to be an *initial split variable* in I if there does not exist another split variable Y s.t. $\mathsf{Y} \prec_{\mathrm{I}} \mathsf{X}$. An initial split variable, X, is said to be *resolved* if there does not exist another variable $\mathsf{Y} \in \mathcal{U}$ that is incomparable with X ($\mathsf{Y} \not\prec_{\mathrm{I}} \mathsf{X}$ and $\mathsf{X} \not\prec_{\mathrm{I}} \mathsf{Y}$), i.e., the information constraints relative to X are resolved/well-defined. This also implies that an SID with a non-empty set of split variables always has an initial split variable, but this split variable is not necessarily resolved/unique. For example, in the SID depicted in Fig. 3, we see that T is the initial split variable and it is also resolved because the remaining variables are observed (possibly never) after deciding on T. On the other hand, in the SID shown in Fig. 4(b), both *BT?* and *UT?* appear as initial split variables but as they are incomparable neither of them are resolved.

Instantiating a split variable, X, corresponds to setting the variable to a specific state, say x, (denoted $\mathsf{X} \mapsto \mathsf{x}$) and evaluating all guards, g that include X in the domain (denoted $\mathsf{g}[\mathsf{X} \mapsto \mathsf{x}]$).[2] If a guard evaluates to false given the instantiation of X, then the associated structural arc is removed together with all the variables that we can only "encounter" (or reach) by following that arc.

Definition 4.2 *A node* Y *is said to be* reachable *from* X *in* I *given* $\mathsf{X} \mapsto \mathsf{x}$ *if either:*

- X *and* Y *are incomparable (i.e.,* $\mathsf{X} \not\prec \mathsf{Y}$ *and* $\mathsf{Y} \not\prec \mathsf{X}$*) or*
- *there is a structural arc* $(\mathsf{X}, \mathsf{Z}) \in \mathcal{E}_{\mathrm{I}}$*, where* $\mathsf{g}_{(\mathsf{X},\mathsf{Z})}[\mathsf{X} \mapsto \mathsf{x}] \equiv$ *true and either:*

[2] Note that after the guards have been evaluated X is no longer a split variable.

- $Z = Y$ *or*
- *there is a path* $(Z = Z_1, \ldots, Z_m = Y)$ *with only structural arcs, where* $g_{(Z_i, Z_{i+1})}[X \mapsto x] \not\equiv$ *false, for all* $1 \leq i \leq m-1$.

Thus, instantiating a split variable in an SID I may effectively remove a subset of the variables and arcs in I. This also implies that we can interpret the instantiation of a split variable, X, in I as reducing I to another SID I' (also denoted $I[X \mapsto x]$), which can be defined based on the observed variables in the past of X as well as the observed variables reachable from X given the instantiation.

Example 4.1 *Consider the SID representation of the* Reactor problem *shown in Fig. 3, where T appears as the initial split variable. By instantiating T w.r.t. T* $\mapsto$ *t and T* $\mapsto$ *nt we obtain the reduced SIDs shown in Fig. 6(a) and Fig. 6(b), respectively.*

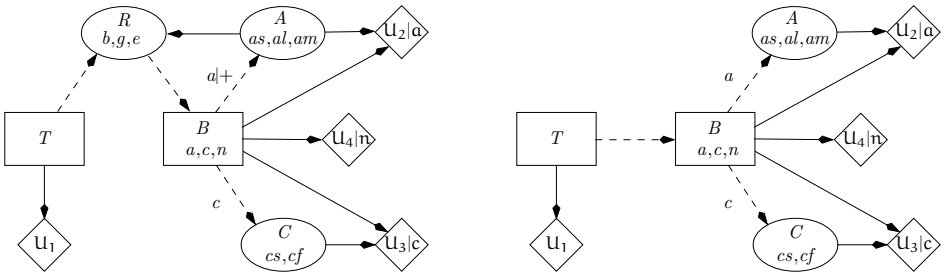


Fig. 6. Figure (a) shows the SID $I[T \mapsto t]$ obtained from the SID representation of the Reactor problem (depicted in Fig. 3) by the instantiation $T \mapsto t$; the + denotes that the choice $B = a$ is only allowed in scenarios that satisfy $(R = e \vee R = g)$. Figure (b) shows the SID $I[T \mapsto nt]$ obtained by the instantiation $T \mapsto nt$; observe that the choice of $B = a$ is not constrained.

Note that R is the initial split variable in $I[T \mapsto t]$ *whereas B is the initial split variable in* $I[T \mapsto nt]$ *(both split variables are resolved).*

More formally, by instantiating a split variable X in an SID I we *reduce* I to another SID $I' = I[X \mapsto x]$ defined as:[3]

- $\mathcal{U}^o_{I'} = \{Y \in \mathcal{U}^o_I | Y \prec X\} \cup \{Y \in \mathcal{U}^o_I | Y$ is reachable from X given $(X \mapsto x)\}$.
- $\mathcal{U}^u_{I'} = \mathcal{U}^u_I$.
- $\mathcal{E}_{I'} = \{(Y, Z) \in \mathcal{E}_I | \{Y, Z\} \subseteq \mathcal{U}^u_{I'} \cup \mathcal{U}^o_{I'} \wedge g_{(Y,Z)}[X \mapsto x] \not\equiv \text{false}\}$.
- $\mathcal{G}_{I'} = \{g[X \mapsto x] | g \in \mathcal{G}_I \wedge g[X \mapsto x] \not\equiv \text{false}\}$.
- $\Phi_{I'} = \{\phi_Y(X = x) | \phi_Y \in \Phi_I \wedge Y \in \mathcal{U}^u_{I'} \cup \mathcal{U}^o_{I'}\}$.

[3] We shall assume that neither I nor $I[X \mapsto x]$ contain barren variables [16]; otherwise they are simply removed. Note that the definition of a barren variable easily carries over to SIDs.

- $\Psi_{I'} = \{\psi_Y(X = x) | \psi_Y \in \Psi_I \wedge Y \in \mathcal{U}^u_{I'} \cup \mathcal{U}^o_{I'}\}$.

Note that if $X \notin \mathrm{dom}\,(\phi_Y)$ then $\phi_Y(X = x) = \phi_Y$ and similar for ψ_Y. Moreover, we shall sometimes refer to $I[X \mapsto x]$ as $I[x]$ if this does not cause any ambiguity.

Since the instantiation of a split variable produces a new SID with another initial split variable we define an *admissible instantiation* of an SID I as an ordered configuration $\mathbf{s} = (S_1 = s_1, \ldots, S_m = s_m)$ over a subset of the split variables in I s.t.:

- S_1 is an initial split variable in I.
- S_i is an initial split variable in $I[S_1 \mapsto s_1] \cdots [S_{i-1} \mapsto s_{i-1}]$ and s_i is a possible state for S_i in $I[S_1 \mapsto s_1] \cdots [S_{i-1} \mapsto s_{i-1}]$.

Given an admissible instantiation $\mathbf{s}$ of I we say that the SID $I[S_1 \mapsto s_1] \cdots [S_m \mapsto s_m]$ is *reduced* from I (or is *reachable* from I). An admissible instantiation $\mathbf{s}$ is said to be *complete* if there does not exist any split variables in $I[S_1 \mapsto s_1] \cdots [S_m \mapsto s_m]$. In what follows we shall use $I[\mathbf{S} \mapsto \mathbf{s}]$ (or simply $I[\mathbf{s}]$) as a shorthand notation for $I[S_1 \mapsto s_1] \cdots [S_m \mapsto s_m]$

Based on the definitions above, we now introduce the concept of a decomposition graph, which is used as the computational structure when solving an SID. The decomposition graph is basically constructed by following the possible temporal orderings and recursively instantiating the split variables w.r.t. their possible states. The recursion is guaranteed to terminate since we have a finite number of split variables and we require that each temporal cycle is resolved/broken before we observe or decide upon any of the variables which appear in that cycle.

More formally, we syntactically define a *decomposition graph* as a labeled directed acyclic graph $G = (\mathcal{N}, \mathcal{A})$ with nodes $\mathcal{N}$ and arcs $\mathcal{A}$. Each node $N \in \mathcal{N}$ is associated with a 2-tuple $(\mathcal{F}, \mathcal{S})$, where:

- $\mathcal{F}$ is a subset of the variables in $\mathcal{U}_C \cup \mathcal{U}_D$.
- $\mathcal{S}$ is either a singleton (a split variable) or the empty set.

If $\mathcal{S} \neq \emptyset$ in the node $N = (\mathcal{F}, \mathcal{S})$, then the arcs emanating from N are labeled with the states of $S \in \mathcal{S}$ (denoted $(N, \cdot)_s$). On the other hand, if $\mathcal{S} = \emptyset$ then the arcs are unlabeled.

A node N in a decomposition graph basically represents an SID that can be obtained from the original SID (represented by the root node) through an admissible instantiation of a subset of the split variables as specified by the labeled arcs on the path from the root to N. More specifically, each node $N_I = (\mathcal{F}, \mathcal{S})$ consists of two parts. One part, $\mathcal{S}$, represents the initial split

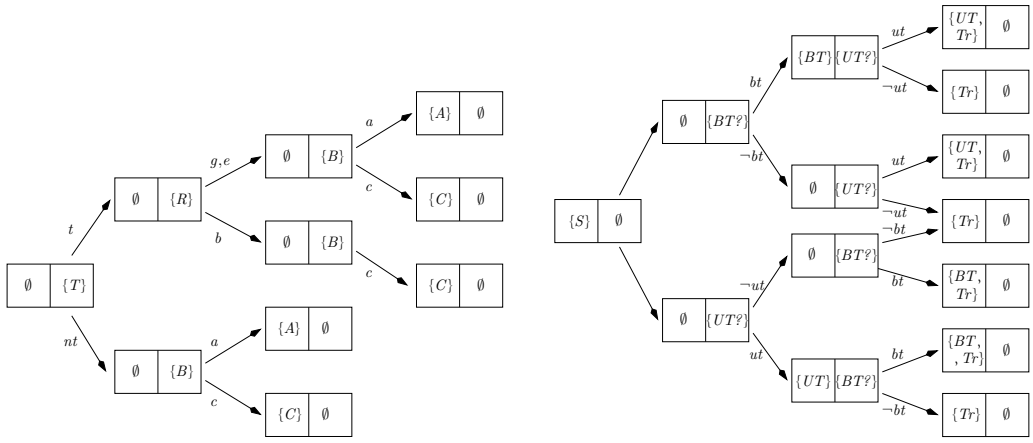
variable (when uniquely defined) of the associated SID I, whereas the other part, $\mathcal{F}$, encodes the nodes (also called the *free variables*) that appear between the split variable in $\mathcal{S}$ and the initial split variable of the SID represented by the parent node of N_{I}. A labeled arc between two nodes N_{I} and $\mathsf{N}_{\mathrm{I}'}$ indicate that the SID I$'$ (represented by the child node) can be obtained from the SID I (represented by the parent node) by instantiating the initial split variable in I according to the label of the arc between the two nodes. For example, in the decomposition graph depicted in Fig. 7(a), the root node represents the SID model I for the REACTOR PROBLEM (see Fig. 3). This node specifies that T is the initial split variable in I ($\mathcal{S} = \{T\}$) and that no other nodes precede T in the temporal order ($\mathcal{F} = \emptyset$). The node, identified by the arc labeled t, encodes that in the SID I$[T \mapsto t]$, R is the initial split variable and there does not exist another node X s.t. $T \prec_{\mathrm{I}[T \mapsto t]} \mathsf{X}$ and $\mathsf{X} \prec_{\mathrm{I}[T \mapsto t]} \mathsf{R}$.

When we are eventually going to solve the SID using the decomposition graph as a computational structure, we also look for an optimal policy for the decision variables that appear as split variables. These policies will obviously depend on the information states for the decision variables involved, hence when an initial split variable is not resolved we should in principle consider all possible refinements of the partial temporal order which can make the information constraints well-defined (and thereby resolve the split variable). More precisely, we define an order refinement as follows:

Definition 4.3 *Let* I *be an SID with a set of unresolved initial split variables $\mathcal{S}$, and let $\mathcal{R}$ be the set of variables that are pairwise incomparable with at least one variable in $\mathcal{S}$ under the partial order $\prec_{\mathrm{I}}$. A minimal extension of the partial order $\prec_{\mathrm{I}}$ over $\mathcal{R} \cup \mathcal{S}$, that resolve a split variable in $\mathcal{S}$, is called an* order refinement *of* I *(denoted* $\mathrm{I}^{\prec'}$*).*

The occurrence of an unresolved initial split variable is encoded in the decomposition graph using unlabeled arcs, i.e., an unlabeled arc between the nodes N_{I} and $\mathsf{N}_{\mathrm{I}'}$ encodes that the initial split variable in I is unresolved and that $\mathsf{N}_{\mathrm{I}'}$ encodes a possible order refinement for I (in the form of the free variables in $\mathsf{N}_{\mathrm{I}'}$). That is, order asymmetry involving split variables is encoded directly in the computational structure. As an example, consider the DIAGNOSIS PROBLEM where *BT?* and *UT?* are both candidates as initial split variables and they are therefore not resolved. In the decomposition graph shown in Fig. 7(b) we see that this aspect is reflected directly in the structure: the root node encodes that S (the symptoms of the patient) are always observed initially, but after this observation the doctor can decide either to perform a blood test (*BT?*) or a urine test (*UT?*).

More formally, in order to construct a decomposition graph for the SID I we invoke the following recursive algorithm (note that unobserved nodes are not taken into account during the construction of the decomposition graph;



(a) (b)

Fig. 7. The left hand figure shows the decomposition graph for the REACTOR PROBLEM. Each node in the decomposition graph is associated with two sets of variables: the variables appearing before the initial split variable in the corresponding decision problem as well as the initial split variable. The right hand figure shows the decomposition graph for the DIAGNOSIS PROBLEM. Notice that the arcs emanating from the root node are unlabeled since the initial split variables are not resolved.

these nodes appear last in the temporal order and they are therefore implicitly associated with the leaf nodes in the decomposition graph):

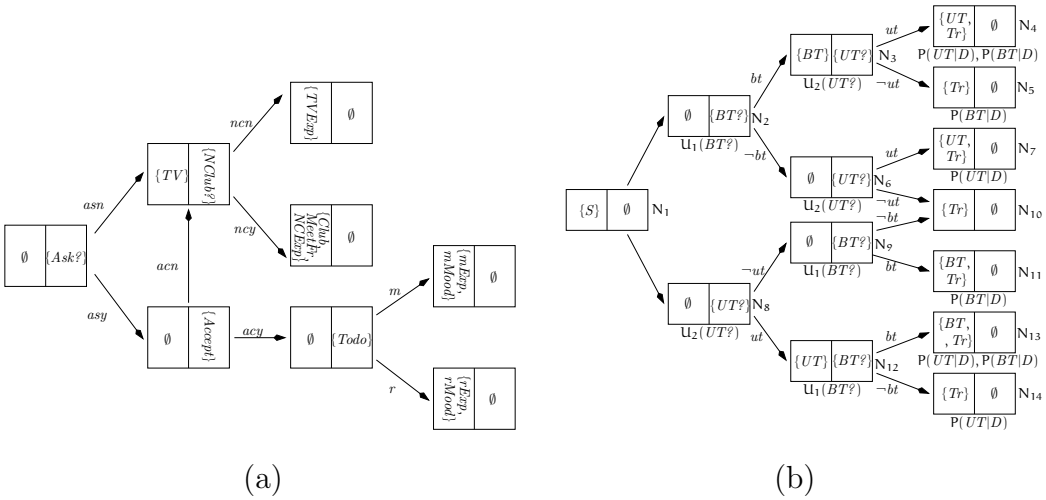
Algorithm (ConstructDecompositionGraph) Let I be an SID with split variables $\mathcal{S}_{\mathrm{I}}$. The corresponding decomposition graph $\mathsf{G}_{\mathrm{I}} = (\mathcal{N}_{\mathrm{I}}, \mathcal{A}_{\mathrm{I}})$ is constructed as follows (observe that we keep track of the nodes, $\mathcal{V}_{\mathrm{I}}$, in I already visited in the previous recursive calls):

1) If $\mathcal{S}_{\mathrm{I}} = \emptyset$ then set $\mathcal{N}_{\mathrm{I}} := \{(\mathcal{U}^{\mathsf{o}} \setminus \mathcal{V}_{\mathrm{I}}, \emptyset)\}$ and return $\mathsf{G}_{\mathrm{I}} = (\mathcal{N}_{\mathrm{I}}, \emptyset)$.
2) If $\mathcal{S}_{\mathrm{I}} \neq \emptyset$ and $\mathsf{S} \in \mathcal{S}_{\mathrm{I}}$ is an initial split variable in I, then:
   a) If S is resolved then
      i) Set $\mathcal{N}_{\mathrm{I}} := \{\mathsf{N} = (\mathcal{F}, \{\mathsf{S}\})\}$, where $\mathcal{F} = \{\mathsf{X} | \mathsf{X} \prec_{\mathrm{I}} \mathsf{S}$ and X is not visited$\}$.
      ii) Mark $\mathcal{F} \cup \{\mathsf{S}\}$ as visited.
      iii) For each $\mathsf{s} \in \mathrm{sp}(\mathrm{S})$
         - Construct a decomposition graph $\mathsf{G}_{\mathrm{I[s]}} = (\mathcal{N}_{\mathrm{I[s]}}, \mathcal{A}_{\mathrm{I[s]}})$ for I[s] by invoking ConstructDecompositionGraph and let N^* be the root of $\mathsf{G}_{\mathrm{I[s]}}$.
         - Set $\mathcal{N}_{\mathrm{I}} := \mathcal{N}_{\mathrm{I}} \cup \mathcal{N}_{\mathrm{I[s]}}$ and $\mathcal{A}_{\mathrm{I}} := \mathcal{A}_{\mathrm{I}} \cup \mathcal{A}_{\mathrm{I[s]}} \cup \{(\mathsf{N}, \mathsf{N}^*)_{\mathsf{s}}\}$.
         - Return $\mathsf{G}_{\mathrm{I}} = (\mathcal{N}_{\mathrm{I}}, \mathcal{A}_{\mathrm{I}})$.
   b) If S is not resolved then
      i) Let $\mathcal{R}$ be the nodes incomparable with an initial split variable in I.[4]
      ii) Set $\mathcal{N}_{\mathrm{I}} := \{\mathsf{N} = (\mathcal{F}, \emptyset)\}$, where $\mathcal{F} = \{\mathsf{X} | \mathsf{X} \prec_{\mathrm{I}} \mathcal{R}$ and X is not visited$\}$.
      iii) Mark $\mathcal{F}$ as visited.

[4] Since S is not resolved we may have more than one initial split variable in I.

iv) For each order refinement $\mathrm{I}^{\prec'}$ of I (see Definition 4.3):
- Construct a decomposition graph $\mathsf{G}_{\mathrm{I}^{\prec'}} = (\mathcal{N}_{\mathrm{I}^{\prec'}}, \mathcal{A}_{\mathrm{I}^{\prec'}})$ for $\mathrm{I}^{\prec'}$ by invoking ConstructDecompositionGraph and let N^* be the root of $\mathsf{G}_{\mathrm{I}^{\prec'}}$.
- Set $\mathcal{N}_{\mathrm{I}} := \mathcal{N}_{\mathrm{I}} \cup \mathcal{N}_{\mathrm{I}^{\prec'}}$ and $\mathcal{A}_{\mathrm{I}} := \mathcal{A}_{\mathrm{I}} \cup \mathcal{A}_{\mathrm{I}^{\prec'}} \cup \{(\mathsf{N}, \mathsf{N}^*)\}$.
- Return $\mathsf{G}_{\mathrm{I}} = (\mathcal{N}_{\mathrm{I}}, \mathcal{A}_{\mathrm{I}})$. □

The algorithm above constructs a decomposition graph with a tree structure that may contain identical substructures. The roots of these substructures basically correspond to the same SID, which can be reached by following different admissible instantiations as encoded in the decomposition graph. Hence to reduce redundancy we can collapse these identical structures. As an example, consider the decomposition graph for the DATING PROBLEM depicted in Fig. 8(a). By instantiating *Ask?* w.r.t. the state *asn* we produce a new decision problem with *NClub?* as the initial split variable, and where the remaining variables are *NClub?*, *TV*, *TVExp*, *Club*, *NCExp* and *MeetFr*. However, an identical SID is produced by the admissible instantiation $\mathit{Ask?} \mapsto \mathit{asy} \wedge \mathit{Accept} \mapsto \mathit{acn}$ (i.e., $\mathrm{I}[\mathit{Ask?} \mapsto \mathit{asy}][\mathit{Accept} \mapsto \mathit{acn}] = \mathrm{I}[\mathit{Ask?} \mapsto \mathit{asn}]$), and the substructures corresponding to these SIDs are therefore collapsed. As we shall see in Section 4.2, this type of merging can be exploited during the evaluation.



(a) (b)

Fig. 8. The left hand figure shows the decomposition graph for the SID representation of the DATING PROBLEM. The right hand figure shows the initialized decomposition graph for the DIAGNOSIS PROBLEM, where the nodes have been labeled from N_1 to N_{14}. In addition to the potentials that are explicitly specified, all of the leaf nodes are also associated with the potentials $\mathsf{P}(D), \mathsf{P}(S|D)$ and $\mathsf{U}_3(\mathit{Tr}, D)$.

Redundancy in the decomposition graph can also come in other forms. For example, we may have two nodes in the decomposition graph that represents SIDs that only differ w.r.t. the ordering over a set of variables of the same type; in this case the SIDs can be considered identical since max-operations commute (similar for sum-operations). This type of redundancy is a consequence of order asymmetry, i.e., decision problems with an unspecified tem-

poral ordering. For instance, consider the decomposition graph for the DIAGNOSIS PROBLEM (shown in Fig. 7(b)). This decomposition graph explicitly encodes admissible extensions of the partially specified temporal ordering, but the nodes corresponding to the SIDs $\mathrm{I}[BT \mapsto \neg bt][UT \mapsto \neg ut]$ and $\mathrm{I}[UT \mapsto \neg ut][BT \mapsto \neg bt]$ are merged because the two SIDs are equivalent. Note also that the decomposition graph does not include e.g. the ordering $BT? \prec UT? \prec BT \prec UT$, since this ordering can be excluded under the assumption of *cost-free* observations. Similarly, we do not consider orderings that can be reached from an ordering, already covered by the decomposition graph, through permutations of neighboring variables of the same type; these points are exploited systematically in [15]. Finally, we note that in the special case where the unspecified temporal order does not involve split variables, the nodes can be considered part of a sub-problem that may be treated as an unconstrained influence diagram, i.e., they will appear as free variables in a single node in the decomposition graph (we shall return to this issue in Section 4.2). As part of future research, we plan to consider algorithms for doing automatic identification (and exploitation) of the types of redundancy discussed above.

4.2 Propagation in decomposition graphs

The decomposition graph is initialized by assigning probability potentials and utility functions (as specified in the SID) to the nodes in the decomposition graph. Starting from the leaves, a node $\mathsf{N}_{\mathrm{I}[\mathbf{S}]} = (\mathcal{F}_{\mathrm{I}[\mathbf{S}]}, \mathcal{S}_{\mathrm{I}[\mathbf{S}]})$ is assigned the potential $\phi \in \Phi_{\mathrm{I}[\mathbf{S}]}$ if $\mathfrak{i})$ ϕ has not already been assigned a node further down the decomposition graph, and $\mathfrak{ii})$ there exists a variable X s.t. $(\mathcal{F}_{\mathrm{I}[\mathbf{S}]} \cup \mathcal{S}_{\mathrm{I}[\mathbf{S}]}) \cap \mathrm{dom}\,(\phi) \neq \emptyset$. For example, Fig. 8(b) shows the decomposition graph for the DIAGNOSIS PROBLEM after it has been initialized with the potentials from the corresponding SID (depicted in Fig. 4(b)); note that in addition to the specified potentials, all the leaf nodes are also implicitly associated with the potentials $\mathsf{P}(D), \mathsf{P}(S|D)$ and $\mathsf{U}_3(Tr, D)$.

Based on the initialization, the decomposition graph can now be solved in almost the same way as an influence diagram [17,18]. For the sake of disposition, we shall for now assume that for any node in the decomposition graph, the free variables do not contain two incomparable variables of different type.

We traverse the decomposition graph by going from the leaves towards the root. When a node is visited we eliminate the associated split variable (if defined) as well as the free variables associated with that node (the elimination is performed in reverse temporal order). The resulting potentials constitute a message which is send upwards in the graph; if a node has several parents, then identical messages are send to all its parents. When a node receives messages from its children, the utility potentials in the messages are combined. More specifically, if the node contains a split variable then the utility potentials are conditioned on the appropriate states of that split variable. On the other hand,

if no split variable is associated with the node, then the utility potentials are combined by maximization, thereby identifying an ordering of the variables which maximizes the expected utility (recall that when an internal node is not associated with a split variable, then its children encode different order refinements of I). Finally, the acyclic probability model, defined by the SID, ensures that the probability potentials in the messages are the same (see also [12]), hence probability potentials are not combined. To illustrate the method, we will show a subset of the calculations that are performed when solving the decomposition graph (depicted in Fig. 8(b)) for the DIAGNOSIS PROBLEM.

Starting with node N_4 we eliminate the variables UT, Tr and D in reverse temporal order:

$$\phi_4^{-D}(S, BT, UT) = \sum_D \mathsf{P}(D)\mathsf{P}(S|D)\mathsf{P}(BT|D)\mathsf{P}(UT|D)$$
$$\psi_4^{-D}(S, BT, UT, Tr) = \frac{1}{\phi_4^{-D}} \sum_D \mathsf{P}(D)\mathsf{P}(S|D)\mathsf{P}(BT|D)\mathsf{P}(UT|D)\mathsf{U}_3(Tr, D)$$

$$\phi_4^{-Tr}(S, BT, UT) = \phi_4^{-D} \text{ and } \psi_4^{-Tr}(S, BT, UT) = \max_{Tr} \psi_4^{-\mathsf{D}}(S, BT, UT, Tr)$$
$$\phi_4^{-UT}(S, BT) = \sum_{UT} \phi_4^{-Tr}(S, BT, UT)$$
$$\psi_4^{-UT}(S, BT) = \frac{1}{\phi_4^{-UT}} \sum_{UT} \phi_4^{-Tr}(S, BT, UT)\psi_4^{-Tr}(S, BT, UT).$$

The resulting potentials (i.e., ϕ_4^{-UT} and ψ_4^{-UT}) are then send upwards to node N_3. Next we eliminate the variables Tr and D in node N_5:

$$\phi_5^{-D}(S, BT) = \sum_D \mathsf{P}(D)\mathsf{P}(S|D)\mathsf{P}(BT|D)$$
$$\psi_5^{-D}(S, BT, Tr) = \frac{1}{\phi_5^{-D}} \sum_D \mathsf{P}(D)\mathsf{P}(S|D)\mathsf{P}(BT|D)\mathsf{U}_3(Tr, D)$$
$$\phi_5^{-Tr}(S, BT) = \phi_5^{-D} \text{ and } \psi_5^{-Tr}(S, BT) = \max_{Tr} \psi_5^{-D}(S, BT, Tr),$$

and send the potentials ϕ_5^{-Tr} and ψ_5^{-Tr} to N_3. When visiting N_3, the messages from N_4 and N_5 are combined by conditioning the utility potentials on the appropriate states of the split variable $UT?$ (note that the probability potentials are identical and therefore unchanged):

$$\psi_3(S, BT, UT?) = \left(\psi_4^{-UT}(S, BT, ut), \psi_5^{-Tr}(S, BT, \neg ut)\right).$$

Afterwards, the variables BT and $UT?$ are eliminated as before. The algorithm then proceeds as above, recursively eliminating the variables in the nodes in the decomposition graph. When reaching N_1 (having no split variable defined), the children N_2 and N_8 are associated with the sets $\{\phi_2^{-BT?}(S), \psi_2^{-\mathrm{B}T?}(S)\}$ and $\{\phi_8^{-UT?}(S), \psi_8^{-UT?}(S)\}$ of potentials, respectively. Analogously to the situation

where a split variable is defined we combine the utility potentials $\psi_2^{-\mathsf{BT?}}(S)$ and $\psi_8^{-UT?}(S)$, however, at $\mathsf{N_1}$ we combine the potentials by taking the maximum for each of their configurations, i.e., $\psi_1(S) = \max\left(\psi_2^{-BT?}(S), \psi_8^{-UT?}(S)\right)$. At this point we also identify the step-function, $\sigma(S)$, associated with S, i.e., the function identifying the ordering of *BT?* and *UT?* that maximizes the expected utility: $\sigma(S = s) = BT?$ if $\psi_2^{-BT?}(S = s) \geq \psi_8^{-UT?}(S = s)$ and *UT?* otherwise. Finally, the optimal policy for any decision variable D can directly be read from the nodes in the decomposition graph having D as a free variable.

In the example above, there did not exist a set of free variables containing a pair of incomparable variables of different type. When such a pair of variables exists, the elimination of the free variables (in the corresponding node in the decomposition graph) does not completely follow the technique for solving influence diagrams but rather the technique for solving unconstrained influence diagrams [15]. That is, we not only look for an optimal policy for the decision variables but also for an optimal set of step functions.

5 Comparison and Discussion

Both AIDs and SIDs use influence diagrams to model preferences and uncertainty, whereas SVNs rely on valuation networks. Thus, the SID model is based on conditional probability tables and allows for chance nodes that are not included in any scenario, thereby supporting the modeler when specifying the probability model; it is often easier to describe such a model using auxiliary variables. This richer model is useful in its own context, but the language of SIDs also allows easy depiction of such larger models. On the other hand, conditional probability tables are not always suitable for domains with a strongly asymmetric structure because they require that the conditioning variables can always co-exist. When this is not the case we may need to either i) augment the state space of the conditioning variables with an artificial state (to ensure co-existence), or ii) to duplicate the head variable so that we have one such variable for each scenario involved.

Analogous to decision trees, SVNs assume that for all scenarios the information constraints are specified as a complete order. If such constraints are only specified up to a partial order, then one has to artificially complete the order during the modeling phase. SIDs use the same underlying structure as SVNs to represent information constraints, but they also allow for clusters of chance and decision variables in order to represent partial temporal orders. Moreover, this construct also enables SIDs to represent order asymmetry which cannot be modeled efficiently using AIDs and SVNs.

Acknowledgments

We would like to thank Manuel Luque and the anonymous reviewers for constructive comments and suggestions for improving the paper.

References

[1] H. Raiffa, R. Schlaifer, Applied Statistical Decision Theory, MIT press, Cambridge, 1961.

[2] R. A. Howard, J. E. Matheson, Influence diagrams, in: R. A. Howard, J. E. Matheson (Eds.), The Principles and Applications of Decision Analysis, Vol. 2, Strategic Decision Group, 1981, Ch. 37, pp. 721–762.

[3] P. P. Shenoy, Valuation-based systems for Bayesian decision analysis, Operations Research 40 (3) (1992) 463–484.

[4] S. M. Olmsted, On representing and solving decision problems, Ph.D. thesis, Department of Engineering-Economic Systems, Stanford University (1983).

[5] H. J. Call, W. A. Miller, A comparison of approaches and implementations for automating decision analysis, Reliability Engineering and System Safety 30 (1990) 115–162.

[6] R. M. Fung, R. D. Shachter, Contingent influence diagrams, Working paper, Department of Engineering-Economic Systems, Stanford University, Stanford, CA (1990).

[7] J. E. Smith, S. Holtzman, J. E. Matheson, Structuring conditional relationships in influence diagrams, Operations Research 41 (2) (1993) 280–297.

[8] R. Qi, N. L. Zhang, D. Poole, Solving asymmetric decision problems with influence diagrams, in: Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, 1994, pp. 491–497.

[9] Z. Covaliu, R. M. Oliver, Representation and solution of decision problems using sequential decision diagrams, Management Science 41 (12) (1995) 1860–1881.

[10] C. Bielza, P. P. Shenoy, A comparison of graphical techniques for asymmetric decision problems, Management Science 45 (11) (1999) 1552–1569.

[11] P. P. Shenoy, Valuation network representation and solution of asymmetric decision problems, European Journal of Operations Research 121 (3) (2000) 579–608.

[12] T. D. Nielsen, F. V. Jensen, Representing and solving asymmetric decision problems, International Journal of Information Technology and Decision Making 2 (2) (2003) 217–263.

[13] L. Liu, P. P. Shenoy, Representing asymmetric decision problems using coarse valuations, Decision Support Systems 37 (1) (2004) 119–135.

[14] R. Demirer, P. P. Shenoy, Sequential valuation networks for asymmetric decision problems, European Journal of Operational Research, In press (2005).

[15] F. V. Jensen, M. Vomlelova, Unconstrained influence diagrams, in: A. Darwiche, N. Friedman (Eds.), Proceedings of the Eightteenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, 2002, pp. 234–241.

[16] R. D. Shachter, Evaluating influence diagrams, Operations Research 34 (6) (1986) 871–882.

[17] F. Jensen, F. V. Jensen, S. L. Dittmer, From influence diagrams to junction trees, in: R. L. de Mantaras, D. Poole (Eds.), Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, 1994, pp. 367–373.

[18] A. L. Madsen, F. V. Jensen, Lazy evaluation of symmetric Bayesian decision problems, in: K. B. Laskey, H. Prade (Eds.), Proceedings of the Fifthteenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers, 1999, pp. 382–390.